**BÖLÜM 27****VERİ TABANI İŞLEMLERİ**

C# veri tabanları üzerinde işlemler gerçekleştirmek üzere DAO, ADO, RDO, ODBC gibi teknolojilere sahiptir. Biz bu bölümde bu teknolojilerden en çok kullanılan ADO.NET’e değineceğiz. ADO .NET ile gerek access, sql, oracle, ibm db2 gibi çeşitli veri tabanları üzerinde kayıt oluşturma , ekleme, güncelleme, sorgulama, silme gibi işlemler kolaylıkla gerçekleştirilebilir.

Veri tabanları üzerinde işlem yapmak için kullanılabilecek veri tabanı nesneleri şunlardır:

**- Connection nesnesi:** Veri tabanına bağlantı kurmak için kullanılır.

**- Command:** Veri tabanı işlemleri ile ilgili kullanılan sorguları çalıştırmak için kullanılır.

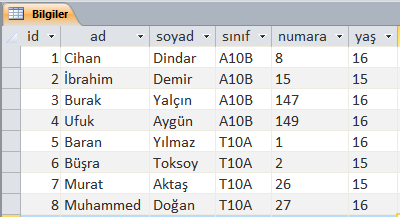
**- DataReader:** Veri tabanı bağlantısı olduğu müddetçe veri tabanı içerisindeki verileri okuma amacı ile kullanılır.

**- DataSet:** Veri tabanı bağlantısı olduğu müddetçe veri tabanı içerisindeki verileri okuma ve yazma amacı ile kullanılır.

**- DataAdapter:** Veri tabanı bağlantısı olmasa bile veriler üzerinde işlemler gerçekleştirilebilir.

**SORGULAR**

SQL sorguları yardımıyla veritabanı içerisindeki tablolar ve kayıtları yöneterek, tablo ve kayıtlar üzerinde düzenlemeler yapabiliriz. SQL diline ait sorguları MSSQL, Access, Oracle, Firebird gibi veri tabanları üzerinde kullanabiliriz. Aşağıdaki gibi örnek bir Access tablosu üzerinde temel SQL sorgularını uygulayarak sonuçlarını görelim.



**- FROM :** Üzerinde işlem yapılacak olan tabloyu belirtir.

**- SELECT :** Veri tabanındaki kayıtları seçmek amacıyla kullanılır.

Örneğin;

SELECT \* FROM Bilgiler

Sql sorgusu ile, Bilgiler tablosundaki bütün kayıtları seçeriz. “\*” karakteri bütün kayıtlar manasına gelmektedir.

SELECT ad,soyad FROM Bilgiler

Sql sorgusu ise, Bilgiler tablosundaki ad ve soyad alanlarındaki tüm kayıtları seçer.



**- DISTINCT :** Belirtilen bir şarta uyan kayıtlardan birden fazla varsa bunlardan yalnızca birinin seçilmesini sağlar.

Örneğin;

SELECT DISTINCT sınıf FROM Bilgiler

Sorgusu sınıf isimlerinden yalnız birer adet listelenmesini sağlar. Dolayısı ile sorgu sonucu



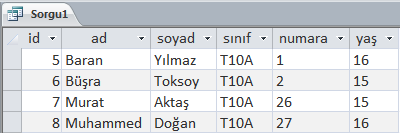
Şeklinde olacaktır.

**- WHERE :** Belirtilen bir şarta göre verileri seçmek amacıyla kullanılır.

Örneğin;

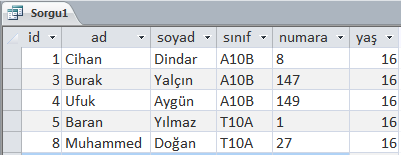
SELECT \* FROM Bilgiler WHERE sınıf= 'T10A'

Sorgusu Bilgiler tablosundaki sınıfı T10A olan kayıtları listelemektedir.



SELECT \* FROM Bilgiler WHERE yaş>15

Sorgusu ise Bilgiler tablosundaki yaşı 15’den büyük olan kayıtları listelemektedir.



**- LIKE :** Bir ya da daha fazla tablo içerisindeki kayıtlar üzerinde karşılaştırma işlemleri yapmak amacıyla kullanılır. Karşılaştırma işlemleri yapılırken joker karakterlerden faydalanılabilir. Kullanılabilecek başlıca karakterler şunlardır:

\* : Birden fazla değeri ifade etmektedir.

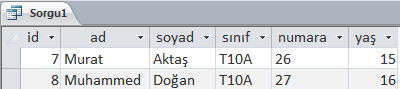
? : Bir tek karakteri ifade etmektedir.

[ ] : Bir değer aralığını ifade etmek amacıyla kullanılır. [A-Z] A ile Z arasındaki harfler gibi.

Örneğin;

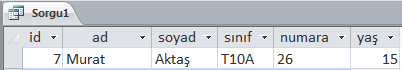
SELECT \* FROM Bilgiler WHERE ad LIKE 'M\*'

Sorgusu Bilgiler tablosundaki adı M ile başlayan kayıtları (Örnek tablomuza göre Murat ve Muhammed kayıtları) listelemektedir.



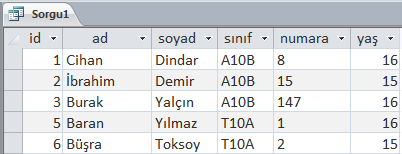
SELECT \* FROM Bilgiler WHERE ad LIKE 'Mu???'

Sorgusu Bilgiler tablosundaki ilk iki harfi “Mu” olan 5 harfli kayıtları listelemektedir. Örnek tablomuza göre şekildeki gibi sadece ad alanında “Murat” yazan değer seçilmiştir.



SELECT \* FROM Bilgiler WHERE ad LIKE '[B-İ]\*'

Sorgusu Bilgiler tablosundaki adları B ile İ harfleri arasındaki harfler ile başlayan kayıtları (Örnek tablomuza göre Burak, Baran, Büşra, Cihan ve İbrahim kayıtlarını) listelemektedir.



**- COUNT :** Belirtilen bir ya da daha fazla tablodaki kayıt sayısını bulmak amacıyla kullanılır.

Örneğin;

SELECT Count(\*) FROM Bilgiler

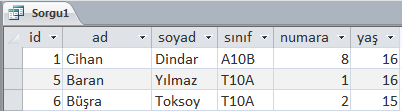
Sorgusu Bilgiler tablosundaki toplam kayıt sayısı olan 8 değerini geriye döndürecektir.

**- BETWEEN :** Bir değer aralığını belirtmek amacıyla kullanılır.

Örneğin;

SELECT \* FROM Bilgiler WHERE numara BETWEEN 1 AND 8

Sorgusu Bilgiler tablosundaki numarası 1 ile 8 arasında olan kayıtları listelemektedir. Sql sorgusunun örnek tablomuz üzerinde çalıştırılması sonucunda aşağıdakine benzer bir sonuç elde edilecektir.

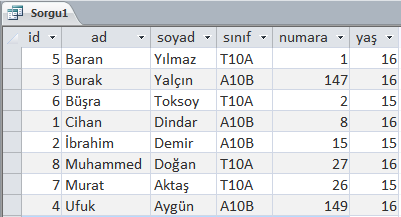


**- ORDER BY :** Kayıtları belirtilen bir alana göre sıralamak amacıyla kullanılır.

Örneğin;

SELECT \* FROM Bilgiler ORDER BY ad

Sorgusu Bilgiler tablosundaki kayıtları ad alanına göre küçükten büyüğe doğru sıralayarak listeler.



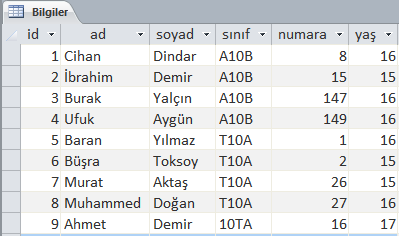
**- INSERT INTO :** Bir tablo içerisinde belirtilen alan ya da alanlara kayıt eklemek amacıyla kullanılır.

Örneğin;

INSERT INTO Bilgiler ( ad, soyad, sınıf, numara, yaş )

VALUES ('Ahmet', 'Demir', '10TA', 16, 17)

Sorgusu Bilgiler tablosundaki ad, soyad, sınıf, numara ve yaş alanlarına VALUES ifadesinden sonra belirtilen kayıtları ekler. Sorgu satırında metin içeren bilgiler tek tırnak içerinde belirtilmelidir. SQL sorgusu çalıştırıldıktan sonra tablomuz aşağıdaki gibi olacaktır.

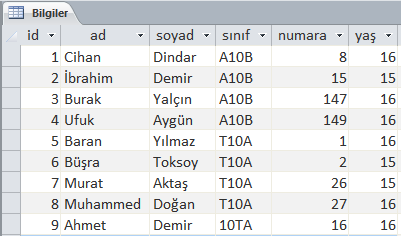


**- UPDATE :** Bir tablo içerisinde belirtilen alan ya da alanlardaki kayıtlar üzerinde değişiklik yaparak güncellemek amacıyla kullanılır.

Örneğin;

UPDATE Bilgiler SET yaş = 16 WHERE ad='Ahmet'

Sorgusu Bilgiler tablosundaki adı Ahmet olan kaydın yaşını 16 olarak değiştirerek kaydı günceller.

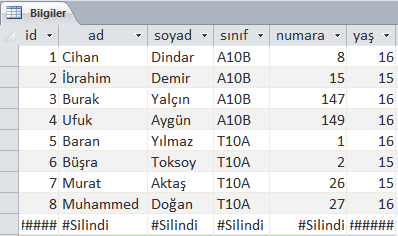


**- DELETE :** Bir tablo içerisinde bir ya da daha fazla kaydı silmek amacıyla kullanılır.

Örneğin;

DELETE FROM Bilgiler WHERE ad='Ahmet'

Sorgusu Bilgiler tablosundaki adı Ahmet olan kaydı siler.



**ACCESS VERİ TABANI İŞLEMLERİ**

Access veri tabanlarına erişim için kullanılabilecek veri tabanı nesneleri şunlardır;

**- OleDBConnection nesnesi**

**- OleDBCommand nesnesi**

**- OleDbDataAdapter**

**- DataReader**

**- DataSet**

Şimdi bu veritabanı nesnelerini kısaca açıklayalım:

**OleDBConnection nesnesi:** Bağlantı açmak ve kapatmak için kullanılan nesnedir. Bu nesne ile veritabanının türü ve veritabanının fiziksel olarak bulunduğu yol belirtilir.

**OleDBCommand nesnesi:** Veritabanı üzerinde SQL deyimleri çalıştırmak için kullanılan nesnedir. Select, Insert, Update ve Delete gibi SQL deyimleri bu nesne üzerinde çalıştırılabilir.

**OleDbDataAdapter:** Connected ve Disconned (bağlantılı ve bağlantısız) özelliği ile veriler arasında köprü görevi görür. Veritabanından çekilen veriyi DataSet’e aktarmak ya da Dataset’te güncellenmiş veriyi veritabanına aktarmada kullanılır.

**DataReader:** Veritabanından kayıt çekmek için kullanılır. Kayıtlar OleDBCommand nesnesinin ExecuteReader metodu ile DataReader’a aktarılır. Bu nesne ile çekilen veriler sırası ile okunabilir.

**DataSet:** Veritabanından bağlantısız olarak çalışan bir nesnedir. OleDbDataAdapter ile veritabanından çekilen veriler DataSet içerisinde tablolar şeklinde saklanır. DataSet’teki veriler üzerinde güncellemeler yapıldıktan sonra OleDbDataAdapter ile tekrar veritabanına aktarılır. DataSet; DataTable, DataColumn, DataRow ve DataRelation nesnelerinden oluşur.

Öncelikle kod yazmadan bir veri tabanına erişimi nasıl sağlarız, onunla ilgili bir uygulama yapalım.

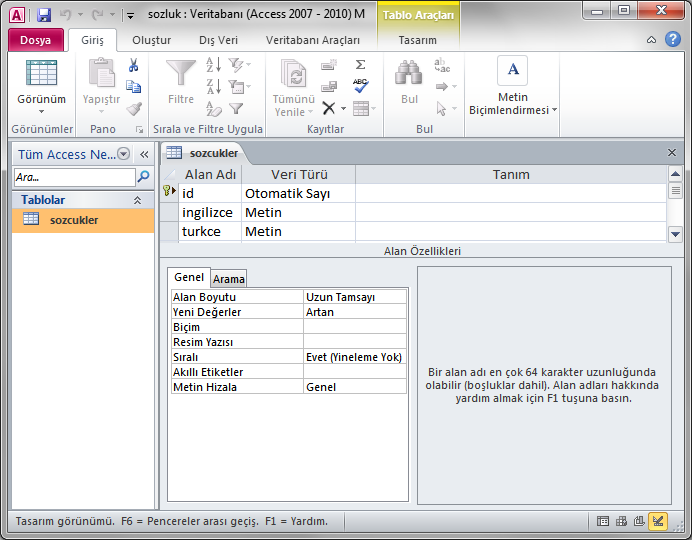
Örnek bir Access veri tabanı oluşturalım.

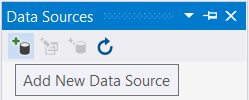
Veri tabanı adı: sozluk

Tablo adı: sozcukler

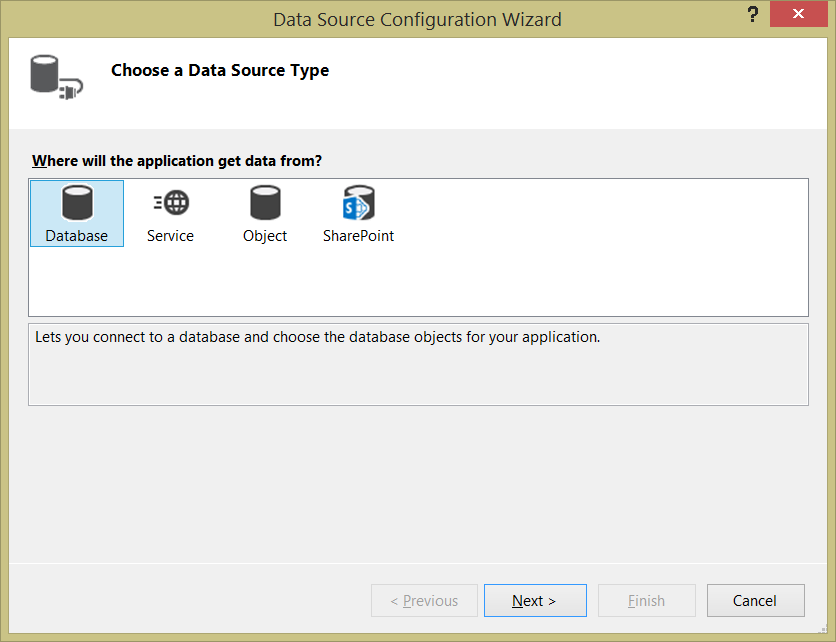
Alan adları: id, ingilizce, turkce

Şeklinde olsun.

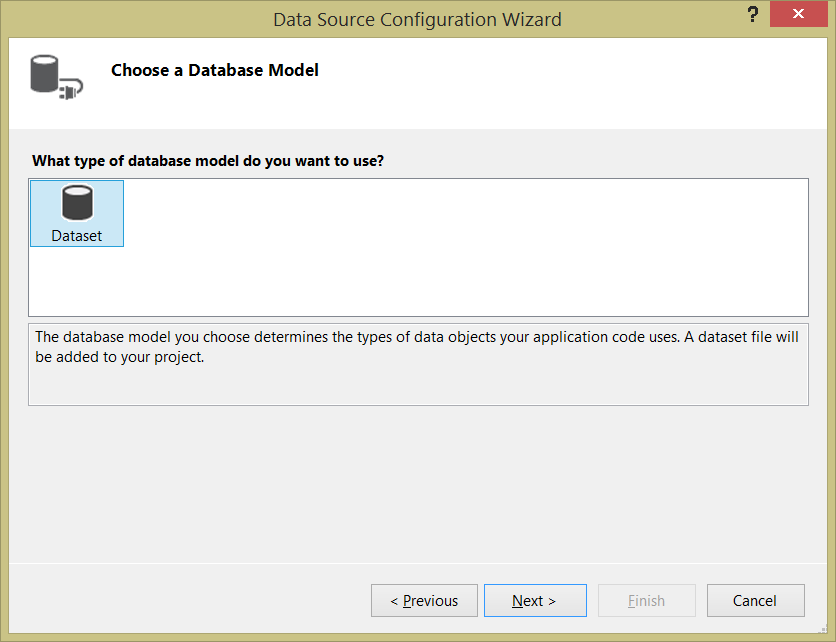


Oluşturmuş olduğumuz veri tabanındaki kayıtları çekebilmek için DataSources penceresini kullanarak yeni bir DataSource oluşturalım. Eğer DataSources penceresi görüntülenmiyorsa View / Other Windows / Data Sources seçenekleri ile görüntüleyebilirsiniz. Bu penceredeki Add New DataSource simgesini tıklıyoruz.

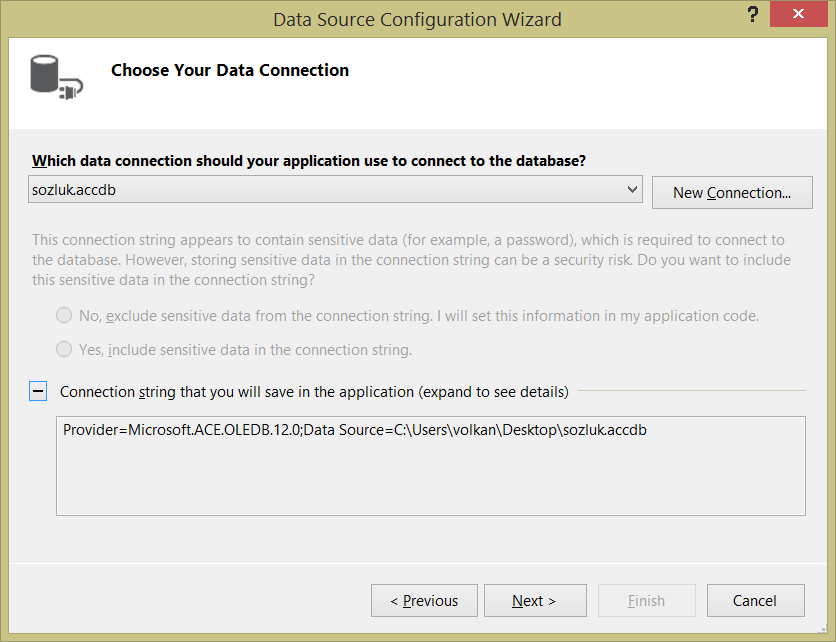
Gelen pencerede Database seçeneğini seçip next butonuna basalım.



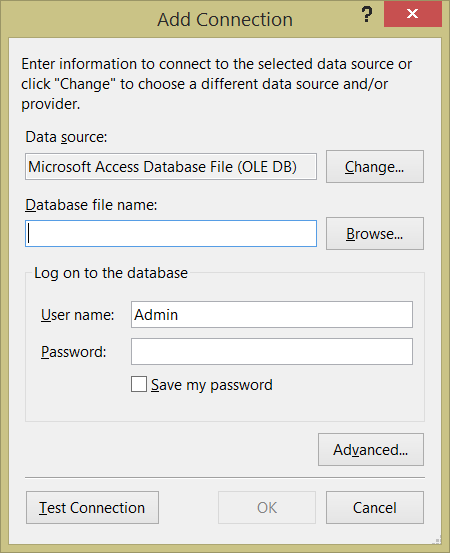
DataSet simgesini seçerek Next butonunu tıklayalım.



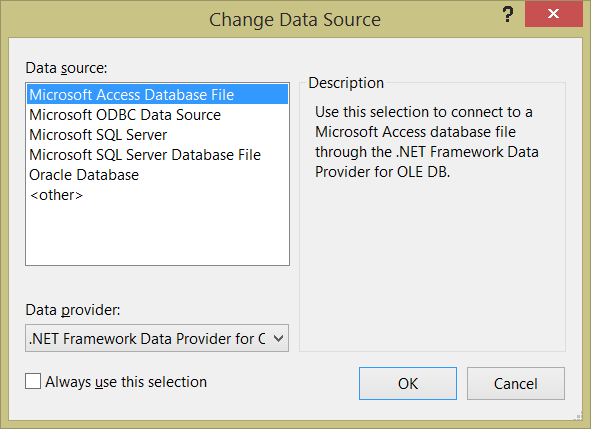
Gelen pencerede daha önce oluşturmuş olduğumuz Access veri tabanımızın ismi hazır olarak eklenmiş olacaktır. Bu pencerede Connection String’in (veri tabanı bağlantı cümlesi) otomatik olarak oluşturulduğunu görebilirsiniz.



Yeni bir veri tabanı bağlantısı oluşturmak için bu penceredeki “New Connection” butonuna tıklıyoruz. Bu pencerede varsayılan olarak “Microsoft Access Database File (OLE DB) seçili durumdadır.

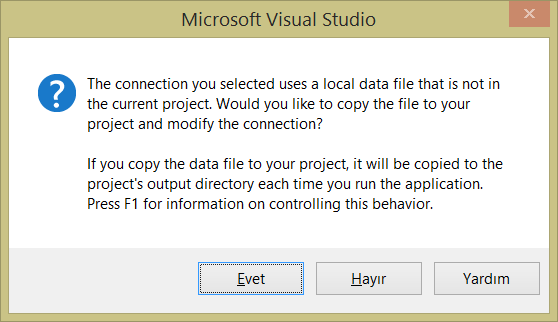


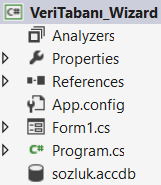
Eğer değiştirmek istersek Change butonuna tıklayarak karşımıza gelen Choose Data Source penceresinden istediğimiz veri tabanı türünü seçerek **OK** butonuna tıklarız.



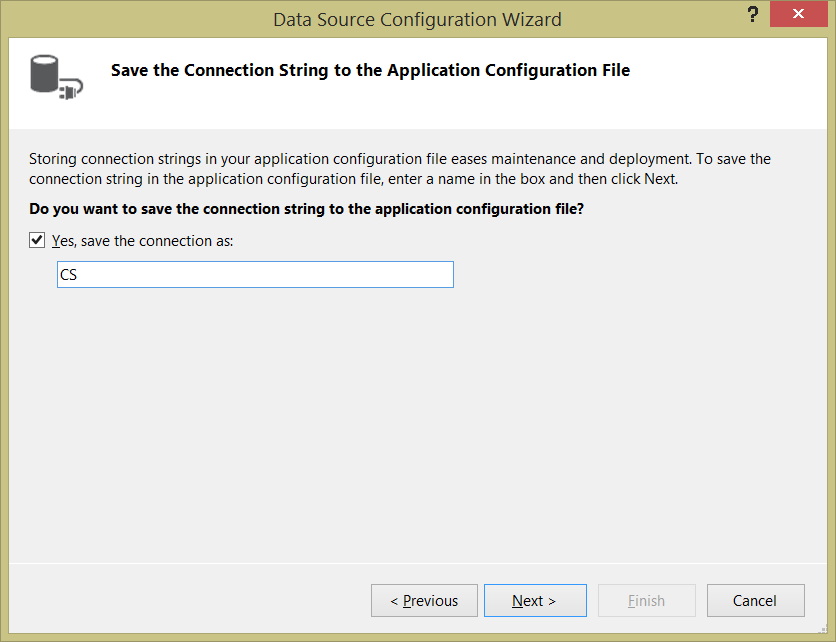
Veri tabanı türünü seçerek **OK** butonuna tıklarız. Eğer veri tabanı için kullanıcı adı ve şifre belirlemişsek “User name” isimli metin kutusuna kullanıcı adı, “Password” isimli metin kutusuna ise veritabanı şifremizi bu bölümde gireriz.

Next butonunu tıkladığımızda veri tabanının bir kopyasının local olarak projemize kopyalanıp kopyalanmamasına dair bir soru gelecektir.

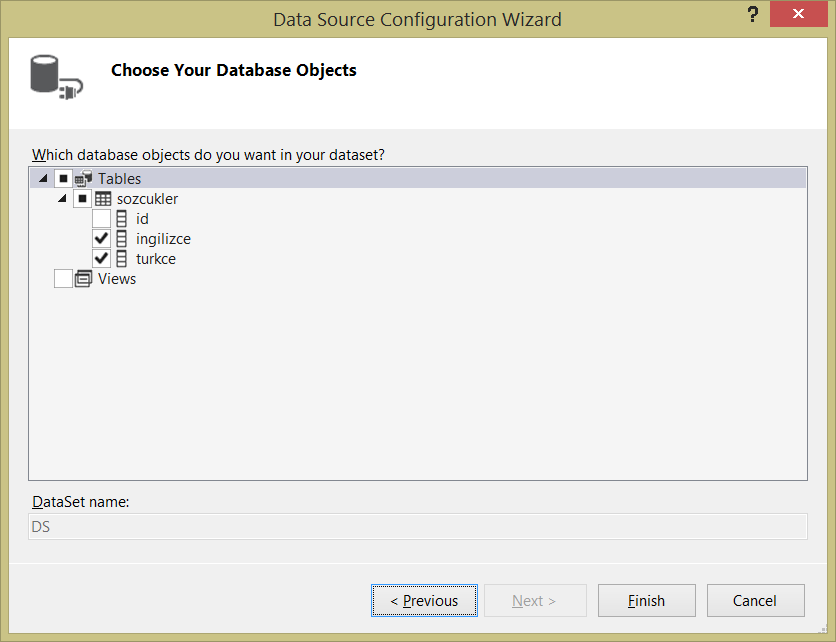


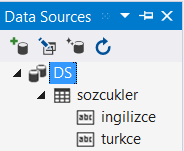
Evet butonuna tıklayarak veri tabanını projemize ekliyoruz.

Gelen pencerede connection string ismini belirliyoruz.

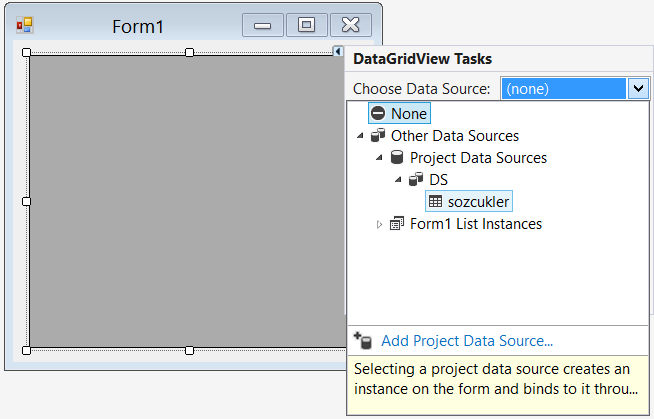


Gerekli ismi verip Next butonuna tıkladığımızda karşımıza dataset içerisinde hangi tablo ve alanların kullanılacağının seçildiği bir pencere gelir. Burada kullanmak istediğimiz tablo ve alanları seçip DataSet ismini belirleyerek Finish butonuna basarak veri tabanı bağlama işlemini bitirmiş oluruz.



Data Sources penceresi içerisine baktığımızda seçtiğimiz tabloların eklendiğini görebiliriz.

Formumuzun üzerine ToolBox içerisinden dataGridView nesnesi ekleyelim. dataGridView’e oluşturmuş olduğumuz dataset’i bağlamak için dataGridView’in sağ üst köşesindeki kulakçığa tıklayarak dataset’i seçelim.



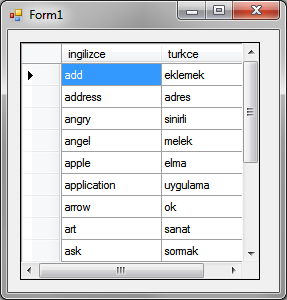
Dataset bağlantı işleminin bitmesinin ardından proje içerisine Dataset, TableAdapter, BindingSource nesnelerinin eklendiğini görürüz.



**Dataset** nesnesi ile, veri tabanı bağlantısının yapılacağı veri tabanı seçilir.

**TableAdapter** nesnesi ile, verilerin çekileceği tablo belirtilir.

**BindingSource** nesnesi ile, veri tabanındaki tabloların içerisinde bulunan alanlar ile form üzerindeki nesnelerin bağlantısı yapılır.

Uygulamamızı çalıştırdığımızda veri tabanındaki değerlerin DataGridView içerisinde listelendiğini görürüz.

**OLEDBCOMMAND**

Veri tabanı işlemleri ile ilgili kullanılan sorguları çalıştırmak için kullanılır. Komutları yürütmek için 4 çeşit Execute metoduna sahiptir:

**- ExecuteReader:** Yürütülen komut sonrasında geriye DataReader nesnesi döndürür.

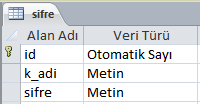
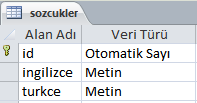
**- ExecuteNonQuery:** Ekleme, silme, güncelleme gibi sorgulardan etkilenen satır sayısını geriye döndürür.

**- ExecuteScalar:** Bu metod geriye tek bir değer döndüren sorgular için kullanılır. Bu değer kayıt sayısı, bir alanın değeri gibi değerler olabilir.

**- ExecuteXMLReader:** Yürütülen komut sonrasında geriye XML Reader nesnesi döndürür.

Veri tabanı işlemlerimizde kullanacağımız örnek bir veri tabanı tasarlayalım. Veri tabanımız bir sözlük uygulamasının veri tabanı olacak. Veri tabanımızın adını “sozluk” olarak belirliyoruz.

Veri tabanımızda“sozcukler” ve “sifre” tablolarını oluşturuyoruz. Bu tablolarda aşağıda görülen alanları oluşturuyoruz.



OleDbConnection, OleDbCommand ve OleDbAdapter nesnelerini kodumuzda kullanabilmek için öncelikle **System.Data** ve **System.Data.OleDb** sınıflarını projemizde dahil etmemiz gerekmektedir.

Daha sonra yeni bir **Connection String (bağlantı cümlesi)** oluşturmamız gerekmektedir.

Access 97-2003 veri tabanları için bağlantı cümlesi aşağıdaki biçimde yazılır:

**OleDbConnection baglantı = new OleDbConnection ("Provider=Microsoft.Jet.Oledb.4.0;Data Source=" +**

**Application.StartupPath + "/sozluk.mdb")**

Access 2007 ve sonrası veri tabanları için ise bağlantı cümlesi aşağıdaki biçimde yazılır:

**OleDbConnection baglantı = new OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + Application.StartupPath + "/sozluk.accdb")**

Access 2007 veri tabanına erişim sağlayabilmek için bilgisayarda Ofis 2007’nin ya da **“2007 Office System Driver: Data Connectivity Components”** isimli bileşenin yüklü olması gerekiyor.

Access 2010 sonrası veri tabanlarına erişim sağlayabilmek için ise bilgisayarda ilgili ofis sürümünün ya da **“Microsoft Access Database Engine 2010”** isimli bileşenin yüklü olması gerekiyor.

**“Application.StartupPath”** deyimi, programın çalıştığı yolu belirtir. Yukarıdaki bağlantı cümlesindeki **“Data Source”** deyimi ile veri tabanımızın; programımız ile aynı klasörde bulunduğunu belirttik.

Veri tabanına bağlantı yaptıktan sonra, yapılacak işlemler için açılması, işlemler sonrasında da kapatılması gerekir. Veritabanını kullanıma açmak için kullanılan kod satırı;

**baglantı.Open( )**

Şeklinde olmalıdır. Yapılan işlemler sonrasında veritabanının kapatılması için kullanılacak kod satırı;

**baglantı.Close( )**

şeklindedir. Veritabanı kapatılmadan tekrar açılmak istendiğinde ise hata mesajı verecektir.

**OleDBConnection** nesnesi ile veri tabanına bağlantı yapıldıktan sonra; kayıt ekleme, kayıt silme ve kayıt güncelleme gibi işlem komutları **Command** nesnesi ile tanımlanmalıdır. Örneğin; Select, Insert, Update, Delete gibi SQL komutları, **Command** nesnesi ile çalıştırılabilir.

Veritabanı ile ilgili işlemler aşağıdaki sıraya uygun olarak yapılmalıdır.

- **OleDbConnection** kullanılarak, veritabanı bağlantı nesnesi tanımlanır.

- Veritabanı üzerinde işlem yapılmak üzere **Open( )** metodu ile kullanıma açılır.

- **OleDbCommand** kullanılarak, yeni bir **Command** nesnesi tanımlanır ve veritabanı üzerinde işlem görecek SQL ifadeleri **Command** nesnesine atanır.

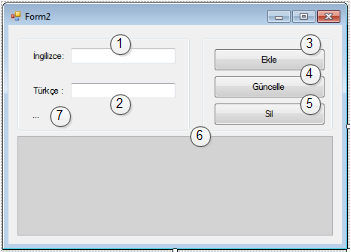
- **Command** nesnesine atanan SQL ifadelerinin çalıştırılması **Insert, Update, Delete** deyimlerinde **ExecuteNonQuery( )** metodu ile, **Select** deyiminde ise **ExecuteReader( )** metodu ile gerçekleştirilir.

- SQL ifadesi ile alınan değerler kullanıldıkdan sonra veritabanı bağlantısı, **Close( )** metodu ile kapatılır.

OleDBCommand kullanımına dair yapacağımız örneğimizde iki adet form bulunacak. İlk form ile şifreli giriş işlemi yapılacak. Eğer şifre doğru girilmiş ise ikinci formumuz olan sözlük kodlarının yer aldığı form ekrana gelecek.

Öncelikle ikinci formumuz olan sözlük kodlarımızın yer aldığı formda gerçekleştirilen veri tabanı işlemlerini sırasıyla inceleyelim.

Formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txt\_ingilizce

(2) txt\_turkce

(3) btnEkle

(4) btnGuncelle

(5) btnSil

(6) dataGridView1

(7) lblKayıtSayısı

**Kayıt Ekleme**

Yeni bir kayıt eklemek için **insert** sorgusunu kullanıyoruz. Insert sorgusu bir tablodaki belirtilen bir alana bir ya da daha çok değeri eklemek amacı ile kullanılır. Ayrıca veri tabanına yeni kayıt eklemek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

OleDbCommand komut = new OleDbCommand();

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("INSERT INTO sozcukler (ingilizce,turkce) VALUES ('" +

txt\_ingilizce.Text + "','" + txt\_turkce.Text + "')",

baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kodumuzda öncelikle baglantı adlı yeni bir connection string (bağlantı cümlesi) tanımladık. Örneğimizde veri tabanı olarak Access 2010 veri tabanı kullanıldığı için bağlantı cümlemizde Microsoft.ACE.OLEDB.12.0 sınıfını kullandık.

Veri tabanındaki kayıtlar üzerinde okuma ya da yazma yapmak için ds isimli yeni bir dataset nesnesi tanımladık.

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

satırında eğer bağlantı durumu kaplı ise veri tabanı bağlantısının açılmasını sağlıyoruz. Aksi takdirde bağlantı açıkken bağlantıyı yeniden açmaya kalktığımızda hata durumu meydana gelecektir.

ds.Clear();

Satırında datasetin içerisini temizliyoruz.

OleDbCommand komut = new OleDbCommand

("INSERT INTO sozcukler (ingilizce,turkce) VALUES ('" +

txt\_ingilizce.Text + "','" + txt\_turkce.Text + "')", baglantı);

satırında OleDbCommand nesnesini kullanarak INSERT sorgusunu çalıştırıp, belirtilen text kutularındaki değerleri veri tabanına ekliyoruz.

komut.ExecuteNonQuery();

satırında OleDbCommand nesnesi içerisindeki sorgumuzun çalıştırılmasını sağlıyoruz. Geriye herhangi bir değer döndürülmeyip sadece sorgu sonucunda kayıt ekleme işlemi gerçekleştirileceği için ExecuteNonQuery metodunu kullandık.

dataGridView1.Update();

Kayıt ekleme işlemi sonrasında dataGridView içerisinde listelenen kayıtların güncellenmesini sağladık.

baglantı.Close();

Veri tabanı bağlantımızı kapattık.

**Kayıt Güncelleme**

Kayıtlar üzerinde güncelleme yapmak için **update** sorgusunu kullanıyoruz. Update sorgusu bir tablodaki belirtilen bir alana ait veri üzerinde yapılan değişiklikler uygulanarak alanda bulunan veri güncellenir. Ayrıca veri tabanındaki kayıtları güncellemek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("UPDATE sozcukler SET ingilizce='" + txt\_ingilizce.Text +

"', turkce='" + txt\_turkce.Text + "' WHERE ingilizce='" +

txt\_ingilizce.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kodumuzda

OleDbCommand komut = new OleDbCommand

("UPDATE sozcukler SET ingilizce='" + txt\_ingilizce.Text +

"', turkce='" + txt\_turkce.Text + "' WHERE ingilizce='" +

txt\_ingilizce.Text + "'", baglantı);

satırında OleDbCommand nesnesini kullanarak UPDATE sorgusunu çalıştırıp, belirtilen text kutularındaki değerler kullanılarak veri tabanındaki kayıtların güncellenmesini sağlıyoruz.

**Kayıt Silme**

Kayıtları silmek için **delete** sorgusunu kullanıyoruz. Delete sorgusu bir tablodaki belirtilen bir alana ait bir ya da daha fazla veriyi silmek amacı ile kullanılır. Ayrıca veri tabanındaki kayıtları silmek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("DELETE FROM sozcukler WHERE ingilizce='" +

txt\_ingilizce.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

dataGridView1.Refresh();

baglantı.Close();

MessageBox.Show("Kayıt Silindi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kodumuzda

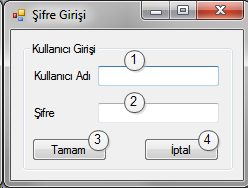
OleDbCommand komut = new OleDbCommand

("DELETE FROM sozcukler WHERE ingilizce='" +

txt\_ingilizce.Text + "'", baglantı);

satırında OleDbCommand nesnesini kullanarak DELETE sorgusunu çalıştırıp, belirtilen text kutusundaki değer veri tabanında aranarak, bulunan kaydın silinmesini sağlıyoruz.

Şifreli giriş formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txtKul\_Adi

(2) txtSifre

(3) btnTamam

(4) btniptal

Şifreli giriş formunun kodu:

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

namespace OleDbCommand\_Sozluk

{

public partial class frmSifre : Form

{

public Form1()

{

InitializeComponent();

}

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

OleDbCommand komut = new OleDbCommand();

string deger;

private void btnTamam\_Click(object sender, EventArgs e)

{

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

komut = new OleDbCommand("SELECT \* FROM sifre where sifre='" +

txtSifre.Text + "' ", baglantı);

try

{

deger = komut.ExecuteScalar().ToString();

if (Convert.ToInt32(deger) == 1)

{

MessageBox.Show("Şifre doğru");

baglantı.Close();

this.Hide();

Form2 frm2 = new Form2();

Frm2.Show();

}

else

MessageBox.Show("Şifre yanlış");

baglantı.Close();

}

Catch { }

}

private void btniptal\_Click(object sender, EventArgs e)

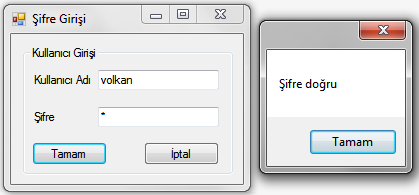
{

Application.Exit();

}

}

}



Şifre giriş formunu yapmaktaki asıl amacımız, **OleDbCommand** nesnesinin **ExecuteScalar** metodunun kullanımını görmektir. Veri tabanımızda tek kullanıcı ve buna ait tek şifre olması nedeni ile **ExecuteScalar** metodunu kullandık. Çünkü **ExecuteScalar** metodu geriye tek bir değer gönderen sorgular için kullanılmaktadır. Eğer **ExecuteScalar** metodu ile çalıştırılan sorgu doğru sonuç verirse geriye **1** değerini göndermektedir. Aksi takdirde **null** değerini göndermektedir. Sorgu doğru sonuç vermediğinde geriye dönen **null** değeri programda hata oluşumuna neden olduğundan dolayı **try catch** bloğu kullandık.

Bağlantı cümlesi tanımlandıktan sonra

**BağlantıCümlesiAdı.Open( )**

Şeklinde veri tabanı bağlantısını açıyoruz. Ancak eğer bağlantı daha önceden açıksa bu durum hata oluşumuna neden olacaktır. Bunu engellemek için programımızda yazdığımız gibi

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

Öncelikle BağlantıcümlesiAdı.State metodu ile bağlantı durumunu kontrol ederek eğer bağlantı durumu ConnectionState.Closed (bağlantı durumu kapalı) ise bağlantıyı açıyoruz.

Kayıt ekleme, güncelleme ve silme işlemlerine dair kodlarımızı birleştirerek sözlük formumuzun kodlarını yazalım.

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

namespace OleDbCommand\_Sozluk

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

OleDbCommand komut = new OleDbCommand();

OleDbDataAdapter adaptor = new OleDbDataAdapter();

DataSet ds = new DataSet();

byte i;

private void Form1\_Load(object sender, EventArgs e)

{

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

komut = new OleDbCommand("SELECT \* FROM sozcukler ", baglantı);

OleDbDataReader reader = komut.ExecuteReader();

reader.Read();

if (reader.HasRows)

{

txt\_turkce.Text = reader["turkce"].ToString();

txt\_ingilizce.Text = reader.GetString(1).ToString();

}

adaptor.SelectCommand = new OleDbCommand

("SELECT ingilizce,turkce FROM sozcukler ",

baglantı);

adaptor.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

komut = new OleDbCommand("SELECT COUNT(\*) FROM sozcukler",

baglantı);

lblKayıtSayısı.Text = "Veritabanında " +

(int)komut.ExecuteScalar() + " tane kelime var.";

baglantı.Close();

}

private void dataGridView1\_CellClick(object sender,

DataGridViewCellEventArgs e)

{

int satirno;

satirno = e.RowIndex;

txt\_ingilizce.Text = dataGridView1.Rows[satirno].

Cells[0].Value.ToString();

txt\_turkce.Text = dataGridView1.Rows[satirno].

Cells[1].Value.ToString();

}

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("INSERT INTO sozcukler (ingilizce,turkce) VALUES

('" + txt\_ingilizce.Text + "','" + txt\_turkce.Text +

"')", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("UPDATE sozcukler SET ingilizce='" +

txt\_ingilizce.Text + "', turkce='" +

txt\_turkce.Text + "' WHERE ingilizce='" + txt\_ingilizce.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

OleDbCommand komut = new OleDbCommand

("DELETE FROM sozcukler WHERE ingilizce='" + txt\_ingilizce.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

dataGridView1.Refresh();

baglantı.Close();

MessageBox.Show("Kayıt Silindi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

}



Form oluşturulurken;

- Sql komutlarını çalıştırmak için **komut** adında **OleDbCommand** nesnesi,

- Veritabanından çekilen veriyi DataSet’e aktarmak ve Dataset’te güncellenmiş veriyi veritabanına aktarmada kullanılmak üzere **adaptor** adında **OleDbDataAdapter** nesnesi,

- **OleDbDataAdapter** ile veritabanından çekilen verilerin **DataSet** içerisinde tablolar şeklinde saklanması ve veritabanı ile ilişkilendirilmesi için **ds** adında **DataSet** nesnesi tanımlıyoruz.

Formun Load olayı esnasında veri tabanındaki kayıtları çekerek DataGridView içerisine ve textbox nesneleri içerisine veri tabanındaki değerleri ekliyoruz.

Kayıtları okumak için **Select** sorgusunu kullanıyoruz. **Select** sorgusu veri tabanındaki kayıtları çekmek amacı ile kullanılır. Kodumuzda

**"SELECT \* FROM sozcukler "**

Sorgusu ile sozcukler tablosundaki bütün kayıtları çektik. **\*** karakteri ile veri tabanındaki bütün kayıtlar çekilir. Aynı sorguyu,

**"SELECT id, ingilizce, turkce FROM sozcukler "**

Şeklinde sozcukler tablosundaki bütün alan isimlerini yazarak da çalıştırabiliriz.

Veri tabanındaki kayıtları çekmek için **OleDbCommand** nesnesinin **ExecuteReader( )** metodunu kullanıyoruz.

**reader.Read( )** metodu ile kayıt çekme işlemini başlatıyoruz.

**“HasRows”** metodu ile sorgumuz çalıştırıldıktan sonra kayıt olup olmadığını sorgulayarak, formun load olayı esnasında DataGridView nesnesinin birinci satırındaki değerleri formdaki textbox nesnelerine yüklüyoruz.

**txt\_turkce.Text = reader["turkce"].ToString( )** satırı ile sorgumuzun çalışması sonucunda oluşan tabloda, ilk satırdaki **turkce** alanında bulunan değeri **txt\_turkce** isimli metin kutusuna ekleniyor.

**txt\_ingilizce.Text = reader.GetString(1).ToString( )** satırı ile de sorgumuzun çalışması sonucunda oluşan tabloda, ilk satırın ikinci sutunundaki (İngilizce karşılığı) değer, **txt\_ingilizce** isimli metin kutusuna ekleniyor. Bu işlemi

**txt\_ingilizce.Text = reader["ingilizce"].ToString( )**

kod satırı ile de gerçekleştirebiliriz.

Veritabanından tüm kayıtları çekip formda kullanmaya hazırlamak için;

**adaptor.SelectCommand = new OleDbCommand**

**("SELECT ingilizce,turkce FROM sozcukler ", baglantı)**

komut satırını kullandık. Adoptor nesnesi ile veritabanımızdan çekilen bilgileri DataSet’e aktarmak için ise

**adaptor.Fill(ds)**

komut satırını kullandık. Bir DataSet’te birden fazla tablo bilgisi tutulabildiğinden dolayı DataSet’teki ilk tablonun formdaki dataGridView’e yükleme işlemini ise

**dataGridView1.DataSource = ds.Tables[0]**

komut satırı ile yapıyoruz.

Ayrıca formun load olayı esnasında veri tabanında bulunan kayıt sayısını bularak görüntülüyoruz. Bu işlem için

**"SELECT COUNT(\*) FROM sozcukler"**

Sorgusunu kullanıyoruz. Bu şekilde sozcukler tablosundaki kayıt sayısını buluyoruz. Geriye tek bir değer dönmesi nedeni ile de **OleDbCommand** nesnesinin **ExecuteScalar( )** metodunu kullanıyoruz.

Kayıt eklemek için kodumuzda

**"INSERT INTO sozcukler (ingilizce,turkce) VALUES ('" + txt\_ingilizce.Text + "','" +**

**txt\_turkce.Text + "')"**

Sorgusu ile sozcukler tablosundaki **ingilizce** ve **turkce** adlı alanlara **txt\_ingilizce** ve **txt\_turkce** adlı text kutularında yazılı olan değerleri aktardık.

Ayrıca veri tabanına yeni kayıt eklemek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

**dataGridView1.Update( )**

komut satırı ile veritabanında yapılan değişikliklerin dataGridView’e aktarılmasını sağlıyoruz.

Kayıtlar üzerinde güncelleme yapmak için kodumuzda

**"UPDATE sozcukler SET ingilizce='" + txt\_ingilizce.Text + "', turkce='" + txt\_turkce.Text +**

**"' WHERE ingilizce='" + txt\_ingilizce.Text + "'"**

Sorgusu ile sozcukler tablosundaki **ingilizce** ve **turkce** adlı alanlardaki veriler, eğer **ingilizce** adlıalandaki veri **txt\_ingilizce** adlı text kutusundaki veriye eşit ise, **txt\_ingilizce** ve **txt\_turkce** adlı text kutularında yazılı olan yeni değerlere göre güncellenir.

Ayrıca veri tabanındaki kayıtları güncellemek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

Kayıtları silmek için kodumuzda

**"DELETE FROM sozcukler WHERE ingilizce='" + txt\_ingilizce.Text + "'"**

Sorgusu ile sozcukler tablosundaki **ingilizce** adlı alandaki veri **txt\_ingilizce** adlı text kutusundaki değere eşit ise bu satırdaki veri silinir.

Ayrıca veri tabanındaki kayıtları silmek için **OleDbCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

C# içerisinde her nesnenin çalışma anında kendine ait olayları (tıklanma, imlecin nesne üzerindeki konumu, üzerine sürükleme gibi) bulunmaktadır. DataGridView’inde kendine has olayları bulunmaktadır. Bu olaylar **DataGridViewCellEventArgs** sınıfının **e** değişkenine aktarılması ile yakalanabilir.

**satirno = e.RowIndex**

komut satırı ile tabloda tıklanan hücrenin satır numarası alınmakta ve

**txt\_ingilizce.Text = DataGridView1.Rows[satirno].Cells[0].Value.ToString( )**

komut satırı ile tıklanan satırın ilk hücresindeki değer, **txt\_ingilizce** isimli text kutusuna aktarılmakta ve Türkçe karşılığı ise,

**txt\_turkce.Text = dataGridView1.Rows[satirno].Cells[1].Value.ToString( )**

kodu ile de **txt\_turkce** isimli text kutusuna aktarılmaktadır.

**OLEDBDATAADAPTER**

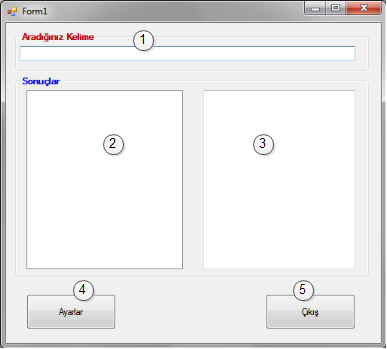
Veri tabanı bağlantısı olmasa bile veriler üzerinde işlemler gerçekleştirilebilir. Veri tabanına bağlanıp veriler çekilir. Daha sonra bağlantı kesilse bile daha önceden çekilmiş veriler üzerinde işlem yapılmaktadır. Üzerinde değişiklik yapılmış veriler daha sonra veri tabanına gönderilir.

OleDbAdapter nesnesini kodumuzda kullanabilmek için öncelikle **System.Data** ve **System.Data.OleDb** sınıflarını projemize dahil etmemiz gerekmektedir.

OleDBDataAdapter kullanımına dair yapacağımız örneğimizde iki adet form bulunacak. İlk formumuz ile sözlük veri tabanımızda arama işlemi yapılacak, bu formdaki Ayarlar butonuna tıklanınca ikinci formumuz olan frmAyarlar isimli form görüntülenecektir. Bu formda ise kayıt ekleme, güncelleme, silme işlemlerini gerçekleştireceğiz.

Öncelikle sözlük veritabanında arama işlemlerinin yapıldığı ilk formumuzda gerçekleştirilen veri tabanı işlemlerini sırasıyla inceleyelim.

Formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txtaranan

(2) lstsonuc

(3) txtsonuc

(4) btnAyar

(5) btnCikis

**Kayıt Listeleme**

Veri tabanındaki kayıtları listelemek için **Select** sorgusunu kullanıyoruz. **Select** sorgusu kullanılarak bir tablodaki belirtilen bir alandaki bir ya da daha çok veriler çekilir ve çekilen veriler **OledbDataAdapter** içerisine doldurulur. Daha sonra çekilen bu veriler datasete aktarılır. Datasetteki veriler de form üzerindeki kontrollere bağlanarak aktarılır.

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void Form1\_Load(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

OleDbDataAdapter adp = new OleDbDataAdapter

("SELECT \* FROM sozcukler", baglantı);

adp.Fill(ds);

lstsonuc.DataSource = ds.Tables[0];

lstsonuc.DisplayMember = "ingilizce";

txtsonuc.DataBindings.Clear();

txtsonuc.DataBindings.Add(new System.Windows.Forms.Binding

("text", ds.Tables[0], "turkce", true, System.Windows.

Forms.DataSourceUpdateMode.OnPropertyChanged));

baglantı.Close();

lstsonuc.SelectedIndex = 0;

txtingilizce.Text = lstsonuc.Text;

txtturkce.Text = txtsonuc.Text;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kayıtların listelenmesi esnasında form üzerindeki kontrollerin veri tabanı bağlantısını kurduk.

Yeni bir dataset tanımlayarak dataadapter nesnesi içerisindeki verileri dataset içerisine aktarıyoruz. Dataset veri tabanından bağlantısız çalışmaktadır. Bir defa bağlanıp veriler çekildikten sonra veri tabanı ile bağlantı kesilir. Dataadapter dataset ile veri tabanında aracılık vazifesi görmektedir. Veri tabanındaki verileri datasete aktarmak, dataset içerisinde güncellenmiş olan verileri de veri tabanına aktarmak amacı ile kullanılır.

listbox nesnesinin veri tabanı ile bağlantısını kurmak için **DataSource** özelliğine dataset içerisindeki verilerden belirtilen tablo ismini çekiyoruz. **Tables[0]** kullanımı ile ilk sıradaki tablo bağlantısı yapılmıştır. Bunun yerine **Tables[“sozcukler”]** şeklinde de kullanılabilir. **DisplayMember** özelliği ile bağlantı kurulacak alan ismi belirtilmiştir.

textBox nesnesinin veri tabanındaki ilgili alanla bağlantısını kurmak için **DataBindings.Add** metodu kullanılabilir.

Örneğimizde görüldüğü gibi txtsonuc adlı textBox nesnesini text veri biçimi ile 0 nolu tablonun (sözcükler tablosuna) turkce adlı alanına bağladık.

**Kayıt Ekleme**

Yeni bir kayıt eklemek için **insert** sorgusunu kullanıyoruz. **Insert** sorgusu bir tablodaki belirtilen bir alana bir ya da daha çok değeri eklemek amacı ile kullanılır. Eklenen veriler önce OledbDataAdapter’e aktarılmakta, buradan da datasete aktarılmaktadır.

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb ");

private void btnEkle\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtingilizce.Text) |

string.IsNullOrEmpty(txtturkce.Text))

{

MessageBox.Show("Lütfen Bütün Alanları Doldurunuz", "Hata");

return;

}

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("INSERT INTO sozcukler (ingilizce,turkce) VALUES ('" +

txtingilizce.Text + "','" + txtturkce.Text + "')", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Eklendi", "Bilgi");

txtingilizce.Text = "";

txtturkce.Text = "";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Güncelleme**

Kayıtlar üzerinde güncelleme yapmak için **update** sorgusunu kullanıyoruz. **Update** sorgusu bir tablodaki belirtilen bir alana ait veri üzerinde yapılan değişiklikler uygulanarak alanda bulunan veri güncellenir. Güncellenen veriler önce OledbDataAdapter’e aktarılmakta, buradan da datasete aktarılmaktadır. Daha sonra veri tabanı ile bağlantı kurulduğunda değişmiş olan veriler, veri tabanındaki eski değerleri ile değiştirilir.

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb ");

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("UPDATE sozcukler SET ingilizce='" + txtingilizce.Text +

"',turkce='" + txtturkce.Text + "' WHERE ingilizce='" +

txtingilizce.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi.", "Bilgi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Silme**

Kayıtları silmek için **delete** sorgusunu kullanıyoruz. **Delete** sorgusu bir tablodaki belirtilen bir alana ait bir ya da daha fazla veriyi silmek amacı ile kullanılır.

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("DELETE FROM sozcukler WHERE ingilizce='" +

txtingilizce.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Silindi","Bilgi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Arama**

Kayıtlar içerisinde arama yapmak için **like** sorgusunu kullanabiliriz. **Like** sorgusu bir tablodaki belirtilen bir alandaki veriler aranan değer ile karşılaştırılarak bulunan değerler geriye döndürülür.

Aşağıdaki örnekte

**"SELECT \* FROM sozcukler WHERE ingilizce LIKE '" + txtaranan.Text + "%'"**

komut satırı ile **txtaranan** isimli metin kutusuna girilen değer ile başlayan kayıtları getiriyoruz.

using System.Data;

using System.Data.OleDb;

public partial class Form1 : Form

{

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void txtaranan\_TextChanged(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

OleDbDataAdapter adp = new OleDbDataAdapter

("SELECT \* FROM sozcukler WHERE ingilizce LIKE '" +

txtaranan.Text + "%'", baglantı);

adp.Fill(ds);

lstsonuc.DataSource = ds.Tables[0];

lstsonuc.DisplayMember = "ingilizce";

txtsonuc.DataBindings.Clear();

txtsonuc.DataBindings.Add(new System.Windows.Forms.Binding

("text", ds.Tables[0], "turkce", true, System.Windows.

Forms.DataSourceUpdateMode.OnPropertyChanged));

baglantı.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

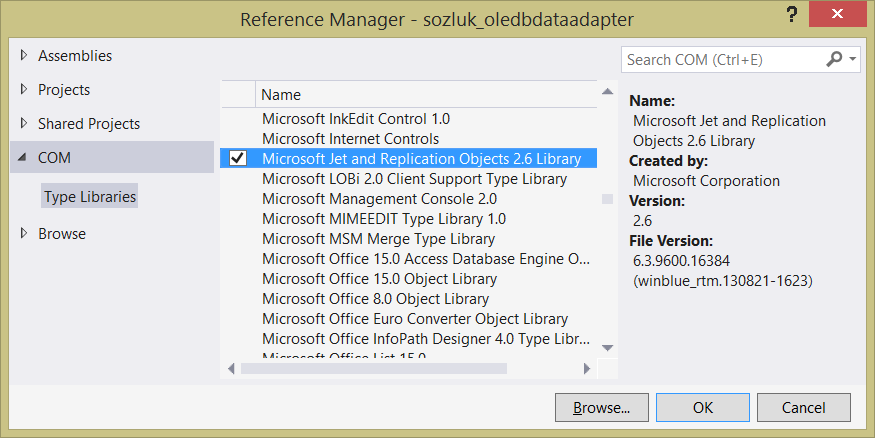
}

}

**Veri tabanını Düzenleme**

Access veritabanları kullanıldıkça kayıt ekleme, silme işlemleri sonucu kayıt sayısına bağlı olmayan boyut artması meydana gelmektedir. Bu da zamanla veritabanına erişimin ve veritabanı üzerindeki işlemlerin yavaşlamasına neden olacaktır. Bunun için veritabanını onaran, hatalarını gideren ve boyutunu küçülten **Compact Database** işleminin uygulanması gerekmektedir.

Öncelikle projemize **“Microsoft Jet and Replication Objects 2.x Library”** adlı dll dosyasını projemize referans olarak eklemeliyiz. Dll dosyasını **Com** sekmesinden seçerek ya da **Browse** butonu ile **“C:\Program Files\Common Files\System\ado\msjro.dll”** dosyasını seçerek ekleyebiliriz.



using System.IO;

public partial class Form1 : Form

{

private void mdb\_duzenle()

{

JRO.JetEngine jro = default(JRO.JetEngine);

jro = new JRO.JetEngine();

jro.CompactDatabase("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb",

"Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/yenisozluk.accdb;" +

"Jet OLEDB:Engine Type=5");

File.Delete(Application.StartupPath + "/sozluk.mdb");

File.Copy(Application.StartupPath + "/yenisozluk.mdb",

Application.StartupPath + "/sozluk.mdb");

File.Delete(Application.StartupPath + "/yenisozluk.mdb");

MessageBox.Show("Veritabanına Bakım ve Düzenleme İşlemi Yapıldı");

}

}

Projemizin iki formdan oluşmakta olduğunu konumuzun başında söylemiştik. İlk formda veri tabanındaki kayıtlar üzerinde arama işlemi gerçekleştirilerek İngilizce sözcükler ve buna karşılık gelen Türkçe sözcükler listelenmektedir. Ayrıca ilk formda veri tabanı üzerinde düzenleme işlemi gerçekleştirilmektedir. Ancak bu işlem her ayın ilk günü bir defa gerçekleştirilecek. frmAyarlar isimli ikinci formda ise kayıt ekleme, güncelleme, silme işlemleri yapılmaktadır.

İlk formumuzun kodu:

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

using Microsoft.Win32;

using System.IO;

namespace OleDbDataAdapter\_Sozluk

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void txtaranan\_TextChanged(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

OleDbDataAdapter adp = new OleDbDataAdapter

("SELECT \* FROM sozcukler WHERE ingilizce LIKE '" + txtaranan.Text + "%'", baglantı);

adp.Fill(ds);

lstsonuc.DataSource = ds.Tables[0];

lstsonuc.DisplayMember = "ingilizce";

txtsonuc.DataBindings.Clear();

txtsonuc.DataBindings.Add(new System.Windows.Forms.Binding

("text", ds.Tables[0], "turkce", true, System.Windows.

Forms.DataSourceUpdateMode.OnPropertyChanged));

baglantı.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnAyar\_Click(object sender, EventArgs e)

{

frmAyar frmayar=new frmAyar();

frmayar.Show();

}

private void btnCikis\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form1\_Load(object sender, EventArgs e)

{

RegistryKey anahtar = Registry.CurrentUser.OpenSubKey

("Software", true);

if (anahtar.OpenSubKey("Sözlük") != null)

{

byte tekrar = Convert.ToByte(anahtar.OpenSubKey

("Sözlük", true).GetValue("CalismaSayisi").ToString());

anahtar.OpenSubKey("Sözlük", true).SetValue("CalismaSayisi",

tekrar + 1, RegistryValueKind.String);

if (DateTime.Now.ToString().Substring(0, 2) == "01")

{

RegistryKey duzenkey = Registry.CurrentUser.OpenSubKey

("Software", true).OpenSubKey("Sözlük", true);

if (duzenkey.GetValue("DuzenlemeZamanı") == null)

{

mdb\_duzenle();

duzenkey.SetValue("DuzenlemeZamanı",

DateTime.Now.ToString().Substring(1, 10));

}

}

else

{

RegistryKey duzenkey = Registry.CurrentUser.OpenSubKey

("Software", true).OpenSubKey("Sözlük", true);

if (duzenkey.GetValue("DuzenlemeZamanı") != null)

{

duzenkey.DeleteValue("DuzenlemeZamanı");

}

}

}

else

{

anahtar.CreateSubKey("Sözlük");

anahtar.OpenSubKey("Sözlük", true).SetValue

("CalismaSayisi", 1, RegistryValueKind.String);

MessageBox.Show("Program Bu Bilgisayarda İlk Defa Çalışıyor");

}

anahtar.Close();

}

private void mdb\_duzenle()

{

JRO.JetEngine jro = default(JRO.JetEngine);

jro = new JRO.JetEngine();

jro.CompactDatabase

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb ",

"Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/yenisozluk.accdb;" +

"Jet OLEDB:Engine Type=5");

File.Delete(Application.StartupPath + "/sozluk.accdb");

File.Copy(Application.StartupPath + "/yenisozluk.accdb",

Application.StartupPath + "/sozluk.accdb");

File.Delete(Application.StartupPath + "/yenisozluk.mdb");

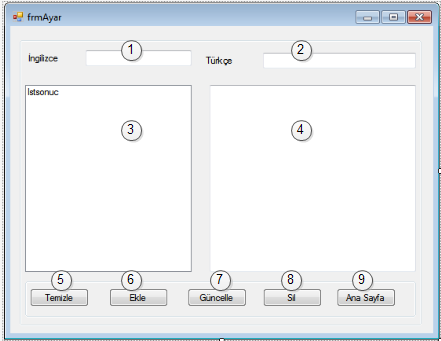
MessageBox.Show("Veritabanına Bakım ve Düzenleme İşlemi Yapıldı");

}

}

}

Kayıt ekleme, güncelleme, silme gibi işlemleri yaptığımız frmAyar isimli ikinci formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txtingilizce

(2) txtturkce

(3) lstsonuc

(4) txtsonuc

(5) btnTemizle

(6) btnEkle

(7) btnGuncelle

(8) btnSil

(9) btnAnasayfa

Formumuzun kodu:

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.OleDb;

namespace OleDbDataAdapter\_Sozluk

{

public partial class frmAyar : Form

{

public frmAyar()

{

InitializeComponent();

}

OleDbConnection baglantı = new OleDbConnection

("Provider= Microsoft.ACE.OLEDB.12.0;Data Source=" +

Application.StartupPath + "/sozluk.accdb");

private void frmAyar\_Load(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

OleDbDataAdapter adp = new OleDbDataAdapter

("SELECT \* FROM sozcukler", baglantı);

adp.Fill(ds);

lstsonuc.DataSource = ds.Tables[0];

lstsonuc.DisplayMember = "ingilizce";

txtsonuc.DataBindings.Clear();

txtsonuc.DataBindings.Add(new System.Windows.

Forms.Binding("text", ds.Tables[0], "turkce", true,

System.Windows.Forms.DataSourceUpdateMode.OnPropertyChanged));

baglantı.Close();

lstsonuc.SelectedIndex = 0;

txtingilizce.Text = lstsonuc.Text;

txtturkce.Text = txtsonuc.Text;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void lstsonuc\_SelectedIndexChanged(object sender, EventArgs e)

{

lstsonuc.SelectedItem.ToString();

txtingilizce.Text = lstsonuc.Text;

txtturkce.Text = txtsonuc.Text;

}

private void btnTemizle\_Click(object sender, EventArgs e)

{

txtingilizce.Clear();

txtturkce.Clear();

}

private void btnEkle\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtingilizce.Text) |

string.IsNullOrEmpty(txtturkce.Text))

{

MessageBox.Show("Lütfen Bütün Alanları Doldurunuz", "Hata");

return;

}

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("INSERT INTO sozcukler (ingilizce,turkce) VALUES ('" +

txtingilizce.Text + "','" + txtturkce.Text + "')", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Eklendi", "Bilgi");

txtingilizce.Text = "";

txtturkce.Text = "";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("UPDATE sozcukler SET ingilizce='" + txtingilizce.Text +

"', turkce='" + txtturkce.Text + "' WHERE ingilizce='" +

txtingilizce.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi.", "Bilgi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

baglantı.Open();

ds.Clear();

OleDbDataAdapter adp = new OleDbDataAdapter

("DELETE FROM sozcukler WHERE ingilizce='" +

txtingilizce.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Silindi","Bilgi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnAnasayfa\_Click(object sender, EventArgs e)

{

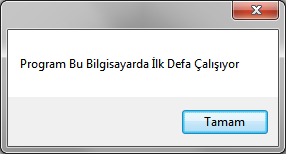
this.Close();

}

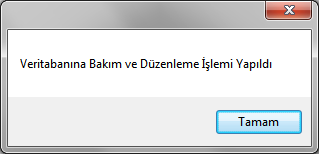
}

}

Programın ilk çalıştırılışı esnasında ekrana “Program Bu Bilgisayarda İlk Defa Çalışıyor” şeklinde bir mesaj penceresi gelecektir.



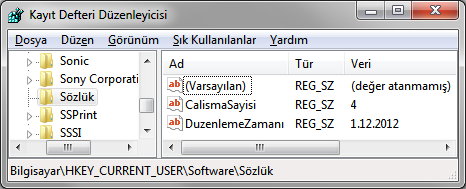
Ayrıca her ayın ilk gününde veri tabanı düzenlenip ekrana “Veri Tabanına Bakım ve Düzenleme İşlemi Yapıldı” şeklinde bir mesaj penceresi gelecektir.



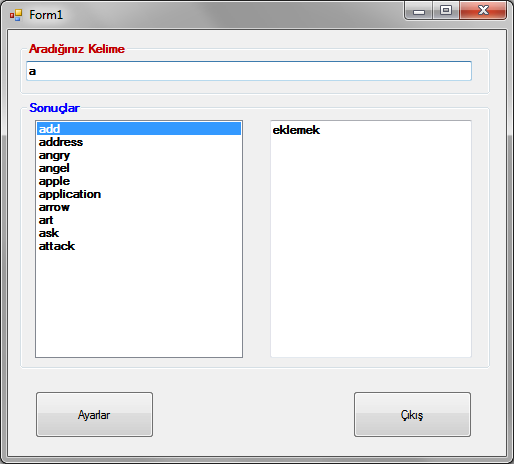
Veri tabanının düzenleme öncesi boyutu 612 kb iken, düzenleme işlemi sonrası 304 kb’a inmiştir.



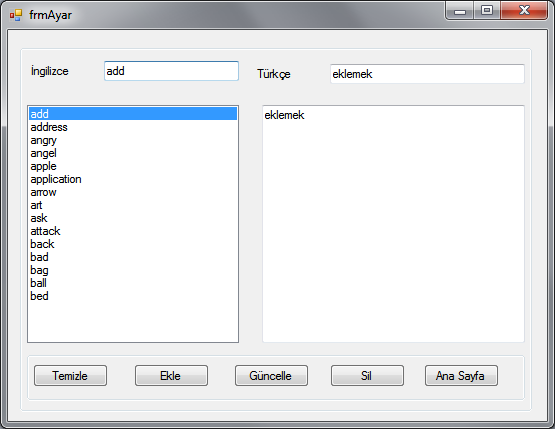
Registry içerisine baktığımızda düzenleme zamanı, çalışma sayısı bilgilerinin kaydedildiğini görürüz.



İlk formumuzda herhangi bir kelimenin ilk harfini yazdığımızda o harfe dair sonuçların listelendiğini görürüz.



Ayarlar butonuna tıklayarak kayıt ekleme , düzenleme ve silme işlemlerini gerçekleştireceğimiz frmAyar adındaki formumuzu görüntüleriz.



**SQL VERİ TABANI İŞLEMLERİ**

SQL veri tabanlarına erişim için kullanılabilecek veri tabanı nesneleri şunlardır;

**- SqlConnection nesnesi**

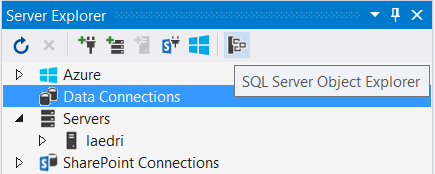
**- SqlCommand nesnesi**

**- SqlDataAdapter**

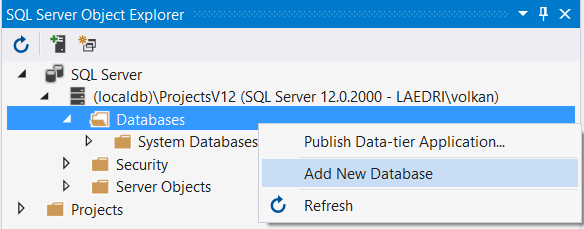
**- DataReader**

**- DataSet**

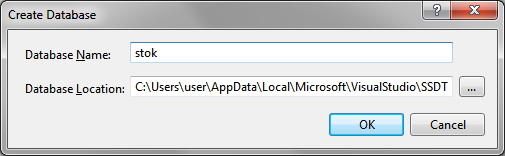
Veri tabanı işlemlerimizde kullanacağımız örnek bir sql veri tabanı tasarlayalım. Veri tabanımız bir stok takip uygulamasının veri tabanı olacak. SQL Server Object Explorer kullanarak SQL veri tabanı oluşturup, içerisinde düzenleme ve kayıt ekleme işlemleri yapabilirsiniz. **Server Explorer** içerisinde SQL Server Object Explorer butonuna tıklayalım.



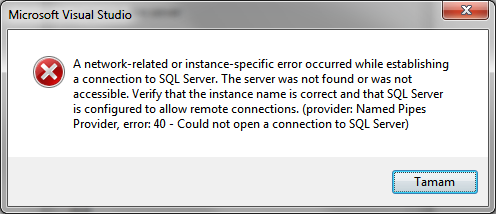
**SQL Server Object Explorer** içerisinde yerel veri tabanları **localdb** başlığı altında yer almaktadır. Biz de sql veri tabanını yerel olarak oluşturacağımız için localdb başlığı altındaki **Databases** ismine farenin sağ tuşu ile tıklayarak gelen menüden “Add New Database” seçeneğini seçelim.



Karşımıza gelen pencerede veri tabanı ismini yazalım.

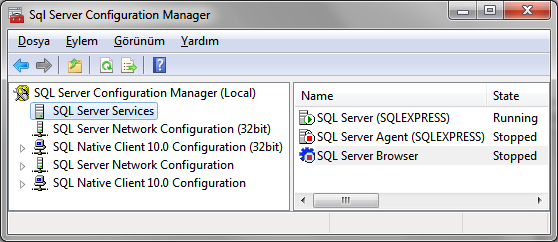


**Not:** Eğer sql veri tabanı oluşturma esnasında remote connection hatası meydana gelirse, nedeni Sql server içerisinde remote connection izni verilmediğinden olabilir.

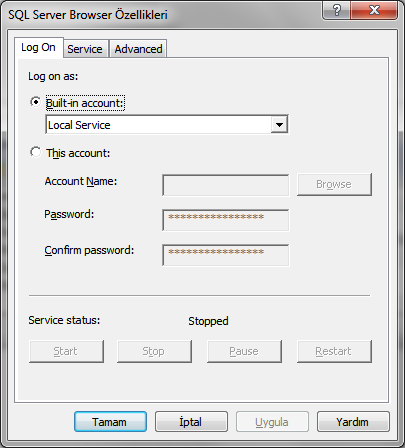


Bu izni vermek içn Başlat / Programlar / Microsoft SQL Server 2008 / Configuration Tools / SQL Server Configuration Manager aracını açıyoruz.

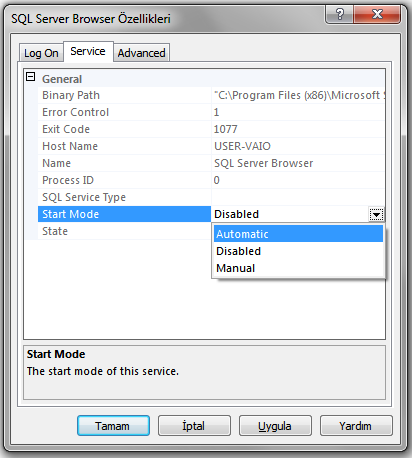
Görüldüğü gibi Sql Server Browser çalışma modunda (Running) değildir. Bunu çalıştıracağız.



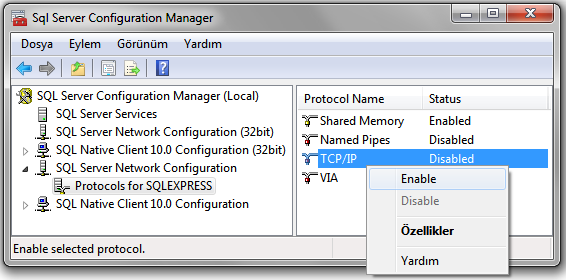
SQL Server Browser ismine çift tıklayarak özellikler penceresini görüntülüyoruz. Log On sekmesinde Built-in account seçeneğinden Local Service seçeneğini seçiyoruz.



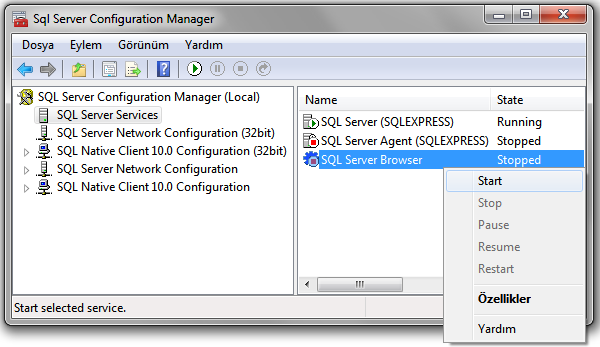
Service sekmesinde Disabled modunda olan Start Mode seçeneğini Automatic olarak ayarlıyoruz.



Özellikler penceresini kapatarak configuration manager penceresinde SQL Server Network Configuration seçeneğine tıklayarak **Protocols for SQLEXPRESS** seçeneğini görüntülüyoruz. Bu seçeneğe tıklayarak sağ panelde **Disabled** konumunda olan **TCP/IP** seçeneğini **Enable** konumuna getiriyoruz.

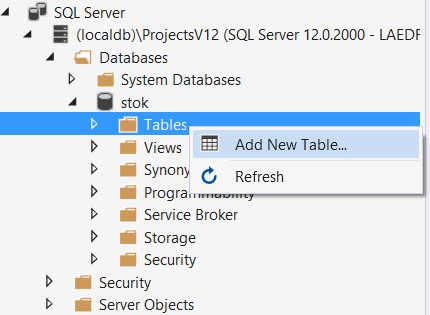


SQL Server Browser’a farenin sağ tuşu ile tıklayarak Start seçeneğini seçip SQL Server Browser’ı başlatıyoruz.

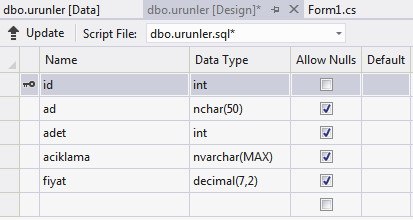


Bu işlemden sonra Sql Server Configuration Manager penceresini kapatarak sql veri tabanı oluşturma işlemini yeniden başlattığımızda hata meydana gelmeden boş bir sql veri tabanı oluşturulacaktır.

SQL Server Object Explorer içerisinde **Databases** bölümünde oluşturmuş olduğumuz veri tabanı ismine tıklıyoruz. Açılan listede **Tables** seçeneğine farenin sağ tuşu ile tıklayarak **Add New Table** seçeneğini tıklayarak veri tabanımızda tablo ve alanları oluşturacağız.



Tablomuzdaki alanları aşağıdaki gibi oluşturuyoruz.

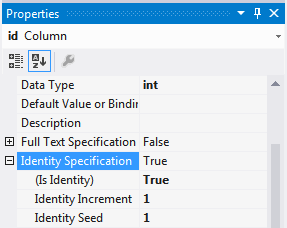


id alanında **Allow Nulls** seçeneğinin işaretini kaldırarak boş değer girilmesini engelliyoruz. Diğer alanları da işaretleyerek boş değer giriş izni veriyoruz.

id alanındaki değerlerin otomatik olarak arttırılması için Properties penceresinde **Identity Specification** seçeneğini Yes olarak seçiyoruz.

**Identity Increment** seçeneği artış değerinin kaçar kaçar olacağını belirtir.

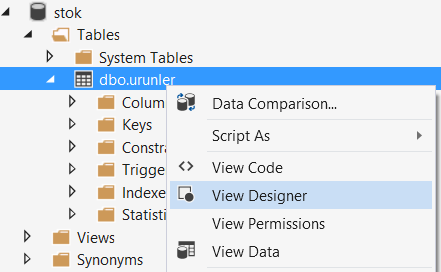
**Identity Seed** seçeneği ise artış işleminin hangi rakamdan itibaren başlatılacağını belirtir.



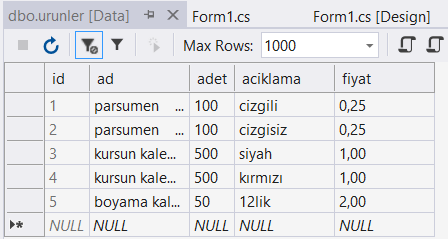
id alanına farenin sağ tuşu ile tıklayarak **Set Primary Key** seçeneğine tıklayıp birincil anahtar olarak belirlenmesini sağlıyoruz.

Tablomuzu kayıt edip urunler ismini veriyoruz. Tüm işlemler bittikten sonra tabloyu güncellemek için Update butonunu tıklıyoruz.

Alanlar üzerinde daha sonra düzenleme yapmak için tablo adına farenin sağ tuşu ile tıklayarak **View Designer** seçeneğini kullanabiliriz.



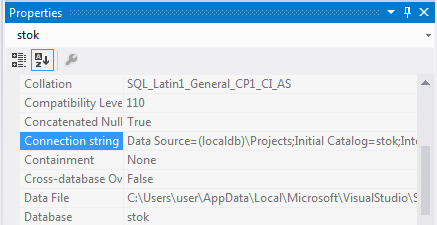
Alanlara kayıt girmek için tablo adına farenin sağ tuşu ile tıklayarak **View Data** seçeneğini kullanabiliriz.



SQLConnection, SQLCommand ve SQLDataAdapter nesnelerini kodumuzda kullanabilmek için öncelikle **System.Data** ve **System.Data.SqlClient** sınıflarını projemizde dahil etmemiz gerekmektedir.

Daha sonra yeni bir **connection string (bağlantı cümlesi)** oluşturmamız gerekmektedir.

Oluşturmuş olduğumuz sql veri tabanı nesnesinin bağlantı cümlesini öğrenmek için SQL Server Object Explorer içerisindeki veri tabanı ismine tıklayıp properties penceresinde Connection String özelliğine bakabiliriz.



Stok örneği için kullanacağımız bağlantı cümlesi şu şekildedir:

**SqlConnection baglantı = new SqlConnection("Data Source=(localdb)\\Projects;Initial Catalog=stok; Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False");**

**SQLCOMMAND**

Sql veri tabanı işlemleri ile ilgili kullanılan sorguları çalıştırmak için kullanılır. OleDbCommand nesnesinde olduğu gibi komutları yürütmek için 4 çeşit execute metoduna sahiptir:

**- ExecuteReader:** Yürütülen komut sonrasında geriye DataReader nesnesi döndürür.

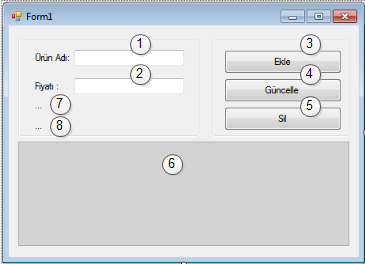
**- ExecuteNonQuery:** Ekleme, silme, güncelleme gibi sorgulardan etkilenen satır sayısını geriye döndürür.

**- ExecuteScalar:** Bu metod geriye tek bir değer döndüren sorgular için kullanılır. Bu değer kayıt sayısı, bir alanın değeri gibi değerler olabilir.

**- ExecuteXMLReader:** Yürütülen komut sonrasında geriye XML Reader nesnesi döndürür.

Oluşturmuş olduğumuz stok veri tabanındaki ürünler tablosunda listeleme, ekleme, günceleme ve silme işlemlerini SqlCommand kullanarak gerçekleştirecek örnek bir uygulama yaparak, gerçekleştirilen veri tabanı işlemlerini sırasıyla inceleyelim.

Formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txt\_ad

(2) txt\_fiyat

(3) btnEkle

(4) btnGuncelle

(5) btnSil

(6) dataGridView1

(7) lblrastgeleurun

(8) lblKayıtSayısı

**Kayıt Listeleme**

Veri tabanındaki kayıtları listelemek için **select** sorgusunu **SqlCommand** nesnesinin **ExecuteReader( )** metodu ile birlikte kullanıyoruz.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30; Encrypt=False;TrustServerCertificate=False");

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

private void Form1\_Load(object sender, EventArgs e)

{

DataSet ds = new DataSet();

if ((baglantı.State == ConnectionState.Closed)) baglantı.Open();

komut = new SqlCommand("SELECT \* FROM urunler ", baglantı);

SqlDataReader reader = komut.ExecuteReader();

reader.Read();

if (reader.HasRows)

{

txt\_ad.Text = reader["ad"].ToString();

txt\_fiyat.Text = reader.GetDecimal(4).ToString();

}

reader.Close();

adaptor.SelectCommand = new SqlCommand

("select ad,fiyat from urunler ", baglantı);

adaptor.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

baglantı.Close();

}

}

textBox nesnelerinin veri tabanı ile bağlantısını kurmak için SQLCommand nesnesinin **ExecuteReader** metodu ile veri tabanından verileri çekiyoruz.  **Reader.Read( )** metodu ile okuma işlemini başlatıyoruz. **Reader.HasRows** metodu ile okunan değerin boş olup olmadığının kontrolü yapılır. **Reader[“ad”]** ve **GetDecimal(4)** metodlarının her ikisi ile birden okunan verilerin textBox içerisine aktarımı yapılır.

**Kayıt Ekleme**

Yeni bir kayıt eklemek için **insert** sorgusunu kullanıyoruz. Ayrıca veri tabanına yeni kayıt eklemek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30; Encrypt=False;TrustServerCertificate=False");

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed) baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("INSERT INTO urunler (ad,fiyat) VALUES ('" +

txt\_ad.Text + "','" + txt\_fiyat.Text + "')", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Güncelleme**

Kayıtlar üzerinde güncelleme yapmak için **update** sorgusunu kullanıyoruz. Ayrıca veri tabanındaki kayıtları güncellemek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30; Encrypt=False;TrustServerCertificate=False");

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed) baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("UPDATE urunler SET ad='" + txt\_ad.Text + "', fiyat='" +

txt\_fiyat.Text + "' WHERE ad='" + txt\_ad.Text +

"'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Silme**

Kayıtları silmek için **delete** sorgusunu kullanıyoruz. Ayrıca veri tabanındaki kayıtları silmek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullanıyoruz.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30; Encrypt=False;TrustServerCertificate=False");

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed) baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("DELETE FROM urunler WHERE ad='" + txt\_ad.Text +

"'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

dataGridView1.Refresh();

baglantı.Close();

MessageBox.Show("Kayıt Silindi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kayıt ekleme, güncelleme ve silme işlemlerine dair kodlarımızı birleştirelim.

Uygulamamızın kodu:

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.SqlClient;

namespace SqlCommand\_Stok

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30; Encrypt=False;TrustServerCertificate=False");

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

DataSet ds = new DataSet();

private void Form1\_Load(object sender, EventArgs e)

{

if ((baglantı.State == ConnectionState.Closed))

baglantı.Open();

komut = new SqlCommand("SELECT \* FROM urunler ", baglantı);

SqlDataReader reader = komut.ExecuteReader();

reader.Read();

if (reader.HasRows)

{

txt\_ad.Text = reader["ad"].ToString();

txt\_fiyat.Text = reader.GetDecimal(4).ToString();

}

reader.Close();

adaptor.SelectCommand = new SqlCommand

("SELECT ad,fiyat FROM urunler ", baglantı);

adaptor.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

komut = new SqlCommand

("SELECT TOP 1 ad FROM urunler ORDER BY NEWID()",

baglantı);

lblrastgeleurun.Text = "Günün indirimli ürünü " +

komut.ExecuteScalar();

komut = new SqlCommand("SELECT COUNT(\*) FROM urunler",

baglantı);

lblKayıtSayısı.Text = "Veritabanında " +

(int)komut.ExecuteScalar() + " tane ürün var.";

baglantı.Close();

}

private void dataGridView1\_CellClick(object sender,

DataGridViewCellEventArgs e)

{

int satirno;

satirno = e.RowIndex;

txt\_ad.Text = dataGridView1.Rows[satirno].

Cells[0].Value.ToString();

txt\_fiyat.Text = dataGridView1.Rows[satirno].Cells[1].

Value.ToString();

}

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("INSERT INTO urunler (ad,fiyat) VALUES ('" +

txt\_ad.Text + "','" + txt\_fiyat.Text + "')", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("UPDATE urunler SET ad='" + txt\_ad.Text + "',

fiyat='" + txt\_fiyat.Text + "' WHERE ad='" +

txt\_ad.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

DataSet ds = new DataSet();

if (baglantı.State == ConnectionState.Closed)

baglantı.Open();

ds.Clear();

SqlCommand komut = new SqlCommand

("DELETE FROM urunler WHERE ad='" +

txt\_ad.Text + "'", baglantı);

komut.ExecuteNonQuery();

dataGridView1.Update();

dataGridView1.Refresh();

baglantı.Close();

MessageBox.Show("Kayıt Silindi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

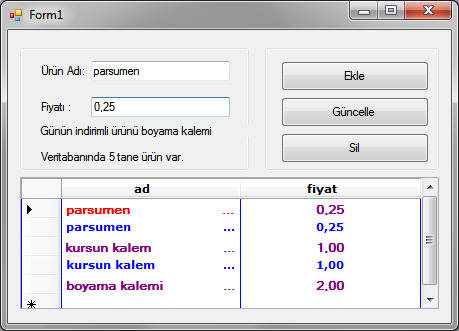
baglantı.Close();

}

}

}

}



Formun Load olayı esnasında veri tabanındaki kayıtları çekerek dataGridView içerisine ve textbox nesneleri içerisine veri tabanındaki değerleri ekliyoruz.

Kayıtları okumak için **Select** sorgusunu kullanıyoruz. **Select** sorgusu veri tabanındaki kayıtları çekmek amacı ile kullanılır. Kodumuzda

**"SELECT \* FROM urunler "**

Sorgusu ile urunler tablosundaki bütün kayıtları çektik. **\*** karakteri ile veri tabanındaki bütün kayıtlar çekilir.

Veri tabanındaki kayıtları çekmek için **SqlCommand** nesnesinin **ExecuteReader( )** metodunu kullanıyoruz.

Ayrıca formun load olayı esnasında veri tabanında bulunan kayıt sayısını bularak görüntülüyoruz. Bu işlem için

**"SELECT COUNT(\*) From urunler"**

Sorgusunu kullanıyoruz. Bu şekilde urunler tablosundaki kayıt sayısını buluyoruz. Geriye tek bir değer dönmesi nedeni ile de **SqlCommand** nesnesinin **ExecuteScalar( )** metodunu kullanıyoruz.

Yine formun Load olayı esnasında rastgele bir ürünü seçerek günün indirimli ürünü olarak görüntülüyoruz. Bu işlem için

**"SELECT TOP 1 ad FROM urunler ORDER BY NEWID( )"**

Sorgusunu kullanıyoruz. Bu şekilde urunler tablosundaki ad alanından rastgele bir değer çekiyoruz. Geriye tek bir değer dönmesi nedeni ile de **SqlCommand** nesnesinin **ExecuteScalar( )** metodunu kullanıyoruz.

Kayıt eklemek için kodumuzda

**INSERT INTO urunler (ad,fiyat) Values ('" + txt\_ad.Text + "','" + txt\_fiyat.Text + "')**

Sorgusu ile urunler tablosundaki **ad** ve **fiyat** adlı alanlara **txt\_ad** ve **txt\_fiyat** adlı text kutularında yazılı olan değerleri aktardık.

Ayrıca veri tabanına yeni kayıt eklemek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

Kayıtlar üzerinde güncelleme yapmak için kodumuzda

**"UPDATE urunler SET ad='" + txt\_ad.Text + "', fiyat='" +**

**txt\_fiyat.Text + "' WHERE ad='" + txt\_ad.Text + "'"**

Sorgusu ile urunler tablosundaki **ad** ve **fiyat** adlı alanlardaki veriler eğer **ad** alanındaki veri **txt\_ad** adlı text kutusundaki veriye eşit ise **txt\_ad** ve **txt\_fiyat** adlı text kutularında yazılı olan yeni değerlere göre güncellenir.

Ayrıca veri tabanındaki kayıtları güncellemek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

Kayıtları silmek için kodumuzda

**"DELETE FROM urunler WHERE ad='" + txt\_ad.Text + "'"**

Sorgusu ile urunler tablosundaki **ad** alanındaki veri **txt\_ad** adlı text kutusundaki değere eşit ise bu satırdaki veri silinir.

Ayrıca veri tabanındaki kayıtları silmek için **SqlCommand** nesnesinin **ExecuteNonQuery( )** metodunu kullandık.

**SQLDATAADAPTER**

Veri tabanı bağlantısı olmasa bile veriler üzerinde işlemler gerçekleştirilebilir. Veri tabanına bağlanıp veriler çekilir. Daha sonra bağlantı kesilse bile daha önceden çekilmiş veriler üzerinde işlem yapılmaktadır. Üzerinde değişiklik yapılmış veriler daha sonra veri tabanına gönderilir.

SQLDataAdapter nesnesini kodumuzda kullanabilmek için öncelikle **System.Data** ve **System.Data.** **SqlClient** sınıflarını projemizde dâhil etmemiz gerekmektedir.

SqlDataAdapter nesnesini SqlCommand sınıfı ile birlikte kullanarak 4 farklı komut biçimi ile seçme, ekleme, güncelleme ve silme işlemlerini gerçekleştirebiliriz.

**SelectCommand:** Kayıtları listelemek amacı ile kullanılır.

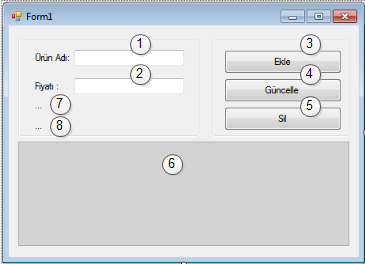
**InsertCommand :** Kayıt eklemek için kullanılır propertydir.

**UpdateCommand:** Kayıtları güncellemek amacı ile kullanılır.

**DeleteCommand:** Kayıt silmek amacı ile kullanılır.

Oluşturmuş olduğumuz stok veri tabanındaki ürünler tablosunda listeleme, ekleme, günceleme ve silme işlemlerini SqlDataAdapter kullanarak gerçekleştirecek örnek bir uygulama yaparak, gerçekleştirilen veri tabanı işlemlerini sırasıyla inceleyelim.

Formumuzun tasarımı aşağıdaki gibi olacak:

 (1) txt\_ad

(2) txt\_fiyat

(3) btnEkle

(4) btnGuncelle

(5) btnSil

(6) dataGridView1

(7) lblenyuksekfiyat

(8) lblendusukfiyat

**Kayıt Listeleme**

Veri tabanındaki kayıtları listelemek için **select** sorgusunu kullanıyoruz. **Select** sorgusu bir tablodaki belirtilen bir alandaki bir ya da daha çok veriyi çekmek amacı ile kullanılır.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30;

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

private void Form1\_Load(object sender, EventArgs e)

{

try

{

adaptor = new SqlDataAdapter

("SELECT ad,fiyat FROM urunler", baglantı);

ds = new DataSet();

adaptor.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

txt\_ad.DataBindings.Clear();

txt\_ad.DataBindings.Add(new System.Windows.Forms.

Binding("text", ds.Tables[0], "ad", true,

System.Windows.Forms.DataSourceUpdateMode.OnPropertyChanged));

txt\_fiyat.DataBindings.Clear();

txt\_fiyat.DataBindings.Add(new System.Windows.Forms.

Binding("text", ds.Tables[0], "fiyat", true,

System.Windows.Forms.DataSourceUpdateMode.OnPropertyChanged));

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

Kayıtların listelenmesi esnasında form üzerindeki kontrollerin veri tabanı bağlantısını kurduk.

Yeni bir dataset tanımlayarak dataadapter nesnesi içerisindeki verileri dataset içerisine aktarıyoruz. Dataset veri tabanından bağlantısız çalışmaktadır. Bir defa bağlanıp veriler çekildikten sonra veri tabanı ile bağlantı kesilir. Dataadapter data set ile veri tabanında aracılık vazifesi görmektedir. Veri tabanındaki verileri datasete aktarmak, dataset içerisinde güncellenmiş olan verileri de veri tabanına aktarmak amacı ile kullanılır.

Listbox nesnesinin veri tabanı ile bağlantısını kurmak için **DataSource** özelliğine dataset içerisindeki verilerden belirtilen tablo ismini çekiyoruz. **Tables[0]** kullanımı ile ilk sıradaki tablo bağlantısı yapılmıştır. Bunun yerine **Tables[“urunler”]** şeklinde de kullanılabilir

Textbox nesnesinin veri tabanındaki ilgili alanla bağlantısını kurmak için **DataBindings.Add** metodu kullanılabilir.

Örneğimizde görüldüğü gibi txt\_ad ve txt\_fiyat adlı textbox nesnelerini text veri biçimi ile 0 nolu tablonun (urunler tablosuna) ad ve fiyat adlı alanlarına bağladık.

**Kayıt Ekleme**

Yeni bir kayıt eklemek için **insert** sorgusunu kullanıyoruz. **Insert** sorgusunu bir SqlCommand nesnesi ile tanımlayarak oluşturacağımız bir SqlDataAdapter nesnesinin SelectCommand özelliğine aktarırız.

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30;

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

DataSet ds = new DataSet();

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter adaptor = new SqlDataAdapter();

string insertSQL = "INSERT INTO urunler (ad, fiyat) " +

"VALUES (@ad, @fiyat)";

SqlCommand insertCMD = new SqlCommand(insertSQL, baglantı);

adaptor.SelectCommand = insertCMD;

insertCMD.Parameters.Add("@ad", SqlDbType.NVarChar, 15).

Value = txt\_ad.Text;

insertCMD.Parameters.Add("@fiyat", SqlDbType.Float, 15).

Value = txt\_fiyat.Text;

adaptor.Fill(ds, "urunler");

MessageBox.Show("Kayıt eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

Kayıt eklemek için kodumuzda

**"INSERT INTO urunler (ad, fiyat) " + "VALUES (@ad, @fiyat)"**

Sorgusunu kullandık. Bu şekilde urunler tablosundaki ad ve fiyat alanlarına karşılık ad ve fiyat adlı parametrelerin değerlerini aktaracağını belirttik. @ işareti kendisinden sonra gelen ifadenin parametre olacağını belirtir.

Parametrelerimize değer aktarmak için ise **Parameters.Add** metodu kullanılır.

Kullanımı şu şekildedir:

**SqlCommandNesnesi.Parameters.Add**

**("@Parametre", SqlDbType.VeriTürü, Alan uzunluğu).Value = AktarılacakNesne**

**Kayıt Güncelleme**

Kayıtlar üzerinde güncelleme yapmak için **update** sorgusunu kullanıyoruz

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30;

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

DataSet ds = new DataSet();

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

baglantı.Open();

ds.Clear();

SqlDataAdapter adp = new SqlDataAdapter

("UPDATE urunler SET ad='" + txt\_ad.Text + "',

fiyat='" + txt\_fiyat.Text + "' WHERE ad='" +

txt\_ad.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

**Kayıt Silme**

Kayıtları silmek için **delete** sorgusunu kullanıyoruz

using System.Data;

using System.Data.SqlClient;

public partial class Form1 : Form

{

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30;

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

DataSet ds = new DataSet();

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

baglantı.Open();

ds.Clear();

SqlDataAdapter adp = new SqlDataAdapter

("DELETE FROM urunler WHERE ad='" + txt\_ad.Text +

"'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Silindi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

}

Kayıt ekleme,güncelleme ve silme işlemlerine dair kodlarımızı birleştirelim.

SqlCommand konusunda kullandığımız stok adlı sql veri tabanını bu uygulamamızda da kullanacağız.

Uygulamamızın kodu:

using System;

using System.Windows.Forms;

using System.Data;

using System.Data.SqlClient;

namespace sqldataadapter\_stok

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

SqlConnection baglantı = new SqlConnection

("Data Source=(localdb)\\Projects;Initial Catalog=stok;

Integrated Security=True;Connect Timeout=30;

SqlCommand komut = new SqlCommand();

SqlDataAdapter adaptor = new SqlDataAdapter();

DataSet ds = new DataSet();

private void Form1\_Load(object sender, EventArgs e)

{

try

{

adaptor = new SqlDataAdapter

("SELECT ad,fiyat FROM urunler", baglantı);

adaptor.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

txt\_ad.DataBindings.Clear();

txt\_ad.DataBindings.Add(new System.Windows.Forms.

Binding("text", ds.Tables[0], "ad", true,

System.Windows.Forms.DataSourceUpdateMode.

OnPropertyChanged));

txt\_fiyat.DataBindings.Clear();

txt\_fiyat.DataBindings.Add(new System.Windows.Forms.

Binding("text", ds.Tables[0], "fiyat", true,

System.Windows.Forms.DataSourceUpdateMode.

OnPropertyChanged));

baglantı.Open();

komut = new SqlCommand

("SELECT MAX(fiyat) AS maksimum FROM urunler", baglantı);

lblenyuksekfiyat.Text = "En yüksek fiyat " +

komut.ExecuteScalar();

komut = new SqlCommand

("SELECT MIN(fiyat) AS minimum FROM urunler", baglantı);

lblendusukfiyat.Text = "En düşük fiyat " +

komut.ExecuteScalar();

baglantı.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnEkle\_Click(object sender, EventArgs e)

{

try

{

SqlDataAdapter adaptor = new SqlDataAdapter();

string insertSQL = "INSERT INTO urunler (ad, fiyat) " +

"VALUES (@ad, @fiyat)";

SqlCommand insertCMD = new SqlCommand(insertSQL, baglantı);

adaptor.SelectCommand = insertCMD;

insertCMD.Parameters.Add("@ad", SqlDbType.NVarChar, 15).

Value = txt\_ad.Text;

insertCMD.Parameters.Add("@fiyat", SqlDbType.Float, 15).

Value = txt\_fiyat.Text;

adaptor.Fill(ds, "urunler");

MessageBox.Show("Kayıt eklendi!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnGuncelle\_Click(object sender, EventArgs e)

{

try

{

baglantı.Open();

ds.Clear();

SqlDataAdapter adp = new SqlDataAdapter

("UPDATE urunler SET ad='" + txt\_ad.Text + "',

fiyat='" + txt\_fiyat.Text + "' WHERE ad='" +

txt\_ad.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Güncellendi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

baglantı.Close();

}

}

private void btnSil\_Click(object sender, EventArgs e)

{

try

{

baglantı.Open();

ds.Clear();

SqlDataAdapter adp = new SqlDataAdapter

("DELETE FROM urunler WHERE ad='" +

txt\_ad.Text + "'", baglantı);

adp.Fill(ds);

baglantı.Close();

MessageBox.Show("Kayıt Silindi");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

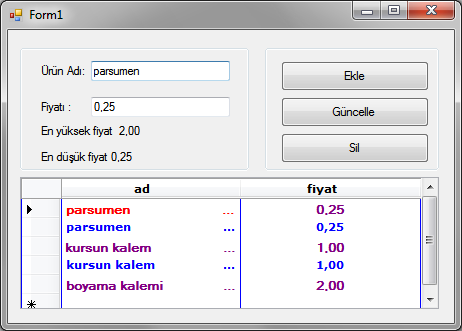
baglantı.Close();

}

}

}

}



Formun Load olayı esnasında veri tabanındaki kayıtları çekerek dataGridView içerisine ve textBox nesneleri içerisine veri tabanındaki değerleri ekliyoruz.

Kayıtları okumak için **Select** sorgusunu kullanıyoruz.

Kodumuzda

**"SELECT ad,fiyat FROM urunler "**

Sorgusu ile urunler tablosundaki ad ve fiyat alanlarını çektik.

Veri tabanındaki kayıtları çekmek için **SqlCommand** nesnesinin **ExecuteReader( )** metodunu kullanıyoruz.

Ayrıca formun load olayı esnasında veri tabanında bulunan kayıt sayısını bularak görüntülüyoruz. Bu işlem için

Formun load olayı esnasında fiyatı en düşük ve en yüksek olan ürünleri seçerek görüntülüyoruz. Fiyatı en yüksek olan ürünü bulmak için **MAX**, en düşük olanı bulmak için **MIN** sorgu deyimi kullanılır.

Bu işlem için örneğimizde

**"SELECT MAX(fiyat) AS maksimum FROM urunler"**

**"SELECT MIN(fiyat) AS minimum FROM urunler"**

sorgularını kullandık. Bu şekilde urunler tablosundaki fiyat alanındaki değerlerden en yüksek ve en düşük olan değerleri çekiyoruz. Geriye tek bir değer dönmesi nedeni ile de **SqlCommand** nesnesinin **ExecuteScalar( )** metodunu kullanıyoruz.

Kayıt eklemek için kodumuzda

**"INSERT INTO urunler (ad, fiyat) " + "VALUES (@ad, @fiyat)"**

Sorgusunu kullandık. Bu şekilde urunler tablosundaki ad ve fiyat alanlarına karşılık ad ve fiyat adlı parametrelerin değerlerini aktaracağını belirttik. @ işareti kendisinden sonra gelen ifadenin parametre olacağını belirtir.

Parametrelerimize değer aktarımını ise

**insertCMD.Parameters.Add("@ad", SqlDbType.NVarChar, 15).Value = txt\_ad.Text**

**insertCMD.Parameters.Add("fiyat", SqlDbType.Float, 15).Value = txt\_fiyat.Text**

kod satırlarında görüldüğü gibi **Parameters.Add** metodu ile yaptık.

Kayıtlar üzerinde güncelleme yapmak için kodumuzda

**"UPDATE urunler SET ad='" + txt\_ad.Text + "', fiyat='" + txt\_fiyat.Text +**

**"' WHERE ad='" + txt\_ad.Text + "'"**

Sorgusu ile urunler tablosundaki **ad** ve **fiyat** adlı alanlardaki veriler eğer **ad** alanındaki veri **txt\_ad** adlı text kutusundaki veriye eşit ise **txt\_ad** ve **txt\_fiyat** adlı text kutularında yazılı olan yeni değerlere göre güncellenir.

Kayıtları silmek için kodumuzda

**"DELETE FROM urunler WHERE ad='" + txt\_ad.Text + "'"**

Sorgusu ile urunler tablosundaki **ad** alanındaki veri **txt\_ad** adlı text kutusundaki değere eşit ise bu satırdaki veri silinir.